"National Research University

Higher School of Economics»

Lyceum

Individual graduation work

IT PROJECT: RPF

https://github.com/Jacob5511/RPF

Performed by: Dergachev Yakov Konstantinovich

Group: 11&3

Moscow – 2023

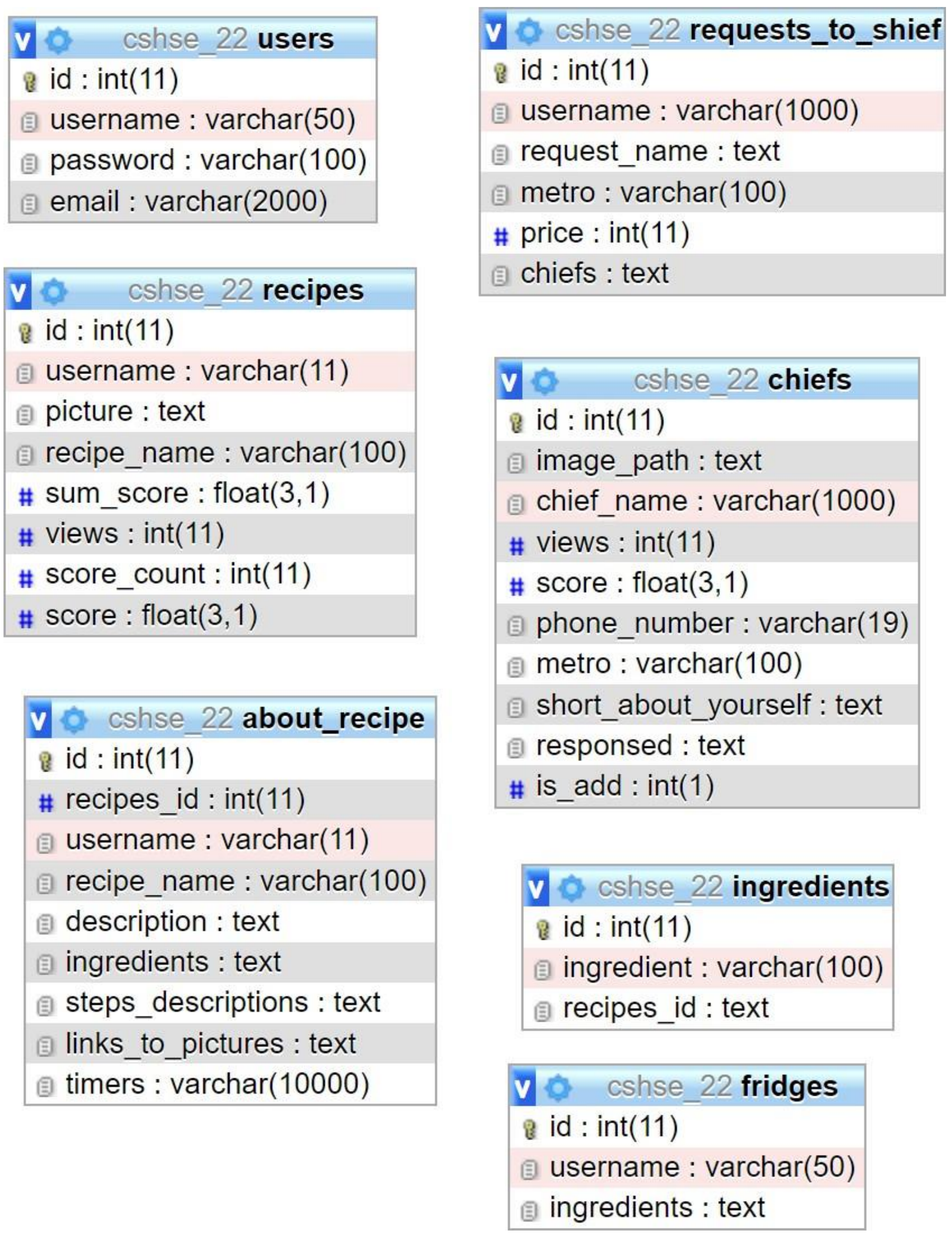
I have limited experience in development. In the past, I created various mini-games, mechanics, and templates for them. I have never worked on full-fledged projects, and I also enjoy programming and creating something new. Because of these factors, I wanted to create something significant, resembling a real project. That’s why I chose an IT project.

It all started when I realized the importance of knowing how to cook at least something. I noticed that more and more people prefer dining in restaurants, cafes, and other establishments instead of cooking at home. This leads to a problem: many forget how to prepare homemade meals, and parents often don’t teach their children how to cook. Because of this, I decided to create an application that would help with cooking.

My application offers a wide range of features. I’ll describe only some of them. My product allows users to view recipes both online and offline, but offline only if the user has saved some recipes beforehand. This can be very convenient if, for example, the diet consists of only certain dishes, users can download the recipes or cooking methods for those dishes and avoid using mobile or home internet data to search for them. It’s also useful for saving internet data, which can be important for many reasons. Additionally, the app allows users to create their own recipes. You don’t have to be a chef, if you know a recipe, why not add it? The app also enables users to post requests for chefs. This helps solve the problem of not knowing how to cook or simply not wanting to. Conversely, the app provides chefs with a way to find clients. Here, a "chef" is someone who can cook the required dishes well. My product also allows users to delete created recipes, filter recipes by multiple criteria, view a chef’s rating, and much more. All these features help address the problems mentioned above.

I managed to implement all user scenarios. The first scenario I implemented was user authentication. My implementation of this mechanic is no different from other standard methods. Users can register if they don’t have an account, log into an existing account, or recover a password via email. The second scenario was the ability to view recipes with or without an internet connection. Users can see recipe information and follow step-by-step instructions for cooking. I tried to make this as user-friendly as possible. I’ve already described the scenarios for ordering a chef and finding clients in the previous paragraph. In the scenario where a client wants to hire a chef, they need to go to the "Find a Chef" page and select a suitable chef. Conversely, if a chef wants to find clients, they can go to the main page for chefs, where all requests are listed, and respond to any of them. If you want to delete a recipe you’ve written, you need to filter all recipes by the "My Recipes" key and delete the desired one. If you have certain ingredients in your fridge, you can sort recipes by the "Recipes I Can Make from My Fridge" key, and the app will show all recipes that include at least one ingredient from your fridge. The app also allows users to edit recipes. To do this, follow the same steps as deleting a recipe but don’t delete it. Next to the "Delete Recipe" button, there’s an "Edit Recipe" button. Clicking it opens the recipe editing menu, where users can change anything: from the main photo to the number of steps and timers in those steps. Additionally, chefs can update their profile information. To do this, log into the chef’s account, go to the chef’s personal cabinet, and click "Edit Profile." This opens the profile editing menu, where everything except the name can be edited: the chef’s photo, brief information, location, and phone number. After making changes, click "Save" to apply them.

Now, I’ll discuss the details of my backend. I didn’t spend much time choosing - I used what I knew. That’s why I chose MySQL. I interact with it either through code or phpMyAdmin. Below is the structure of each table.



cshse_22 users
id : int(11)
username : varchar(50)
password : varchar(100)
email : varchar(2000)

cshse_22 requests_to_shief
id : int(11)
username : varchar(1000)
request_name : text
metro : varchar(100)
price : int(11)
chiefs : text

cshse_22 recipes
id : int(11)
username : varchar(11)
picture : text
recipe_name : varchar(100)
sum_score : float(3,1)
views : int(11)
score_count : int(11)
score : float(3,1)

cshse_22 chiefs
id : int(11)
image_path : text
chief_name : varchar(1000)
views : int(11)
score : float(3,1)
phone_number : varchar(19)
metro : varchar(100)
short_about_yourself : text
responsed : text
is_add : int(1)

cshse_22 about_recipe
id : int(11)
recipes_id : int(11)
username : varchar(11)
recipe_name : varchar(100)
description : text
ingredients : text
steps_descriptions : text
links_to_pictures : text
timers : varchar(10000)

cshse_22 ingredients
id : int(11)
ingredient : varchar(100)
recipes_id : text

cshse_22 fridges
id : int(11)
username : varchar(50)
ingredients : text

I tried to create an optimal number of tables and fields. I believe I succeeded in this task. Now, let's talk about the relationships between these tables—which table is connected to which and by which field. For example, the **recipes** table is linked to the **users** table by the **username** field, and the **about_recipe** table is linked to the **recipes** table by the ***recipes_id*** field. The **chiefs** table is linked to the **users** table by the ***chief_name*** field, and the **fridges** table is linked to the **users** table by the ***username*** field, while the **requests_to_chief** table is linked to the **users** table by the ***username*** field. The **ingredients** table is linked to the **recipes** table by the ***recipes_id*** field.

Before starting development, I faced the question: "What tools should I use?" After some consideration, I settled on the following. For the client side, I used Unity and the C# programming language. For the server side, I used the Flask framework and the Python programming language. For design, I used Adobe Illustrator. I chose these tools because they suited me best. As I mentioned earlier, I had previously worked on small templates and mechanics, and I did that in Unity. My knowledge of Python was very basic, but it was enough for this project.

The development stages were as follows: 1) Developing user scenarios. 2) Designing the interface. 3) Populating the database. 4) Connecting the database. 5) Creating and connecting the server. 6) Launching the Android version of the product. 7) Testing and debugging. 8) Preparing the project for presentation. I managed to meet all deadlines so far. Let’s focus on stages 1, 2, and 3. Developing user scenarios: I wrote the first version of the user scenarios, implemented most of the functions, and then adjusted the scenarios to fit the developed product. Designing the interface: This was probably my least favorite part because I’m not a designer and had never created an app interface before. It took me a long time to account for all possible user actions and "draw" them. But I succeeded. Populating the database: I filled it using a script and pre-prepared data.

Analyzing the work done, I’d like to highlight a few points: 1) Problems I encountered during development and how I solved them. 2) Possible ways to improve or expand the product. 3) Skills I acquired and how I can use them in the future. 4) Risks I faced at the beginning of the project and whether they materialized. Regarding problems, there were quite a few. As I mentioned earlier, I didn’t have much experience in development. For example, one issue was saving images on the server. Images weighing about 50 KB were being saved as 2000 KB. After much thought, I realized the problem wasn’t obvious to me—I was saving them in PNG format instead of JPG. Another problem was saving images taken with a phone camera. Initially, I saved images using their filenames, assuming they were unique. However, camera images had the same name, so they kept overwriting each other. I solved this by generating a unique identifier for each image. There was also an optimization issue—the app froze for a few seconds when updating a recipe. The problem was the same: converting to PNG instead of JPG, which took much longer than necessary. Now, let's discuss point 2. At the moment, I don't plan to further develop this product. However, it could be improved, optimized, and adapted for all Android devices before releasing it on Google Play. Regarding point 3, I acquired skills such as independence, responsibility, time management, and the ability to solve problems faster. These will help me in future projects and in life in general. For point 4, I had several risks: Poor optimization - I partially avoided this; the app's optimization is better than I expected, but it can still be improved. Making the app too heavy (>500 MB) - I avoided this; the app weighs about 40 MB when downloaded and around 100 MB on my phone. Creating an unpleasant interface - I believe I succeeded in making a good, user-friendly interface that doesn't strain the eyes and is pleasant to use.

In conclusion, I did a huge amount of work for myself. I worked on this project for about two months, and it was challenging, but I didn't want to give up or change fields. I gained many new skills that will be useful in the future.